



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Bashant <i>et al.</i>	Conf. No.:	8800
Serial No.:	10/614,968	Art Unit:	2176
Filing Date:	07/08/2003	Examiner:	Shah, Sanjiv
Title:	SYSTEM AND METHOD FOR SYNCHRONIZING RELATED DATA ELEMENTS IN DISPARATE STORAGE SYSTEMS	Docket No.:	END920000147US2 (IBME-0001DIV)

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. 1.131

We, the Applicants in the above-identified patent application, declare as follows:

1. That we are the inventors of the subject matter described and claimed in the above-identified patent application.

2. That prior to August 14, 2000, we conceived of a system having a table of keys for synchronizing related data elements between a first and second storage system, each key comprising:

a universal identifier corresponding to a data element in the first and second storage system;

a first record identifier corresponding to the data element in the first storage system; and

a second record identifier corresponding to the data element stored in the second storage

system, wherein the universal identifier, the first record identifier and the second record identifier are used to synchronize the data element between the first and second storage system.

3. That prior to August 14, 2000, we conceived of a system for synchronizing related data elements between a first and second storage system, comprising:

a header reading system for receiving an instruction from the first storage system, wherein the instruction has a first header that includes a first identifier;

a table interface for accessing a table to identify a second identifier based on the first identifier;

a header generation system for generating a second header corresponding to the second storage system; and

an instruction passing system for passing the instruction and the second header to the second storage system.

4. That the systems are described in a Functional Specification (Exhibit “A”) drafted prior to August 14, 2000, specifically, Section 2 (pages 8-9) and Appendix C (page 22) describe cross referencing and synchronizing data and Section 1 (page 7) describes the message header.

5. That the synchronizing invention is further described in a Disclosure of Invention (Exhibit “B”) submitted to the IBM Corporation Patent Department on August 14, 2000.

6. That, subsequent to the conception of the synchronizing invention, and up until the patent application filing date of October 25, 2000, we diligently and actively assisted the IBM Corporation Patent Department in the planning, preparation, review, and filing of the above-identified patent application.

Declarants further state that the above statements were made with the knowledge that

Declarants further state that the above statements were made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

Date:

8/17/2004


David M. Bashant

Date:

Donald E. Buddenbaum

Date:

Michael L. Denny

Date:

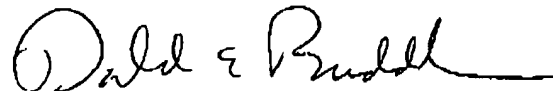
Daniel M. Yellin

Declarants further state that the above statements were made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

Date:

David M. Bashant

Date:



Donald E. Buddenbaum

Date:

Michael L. Denny

Date:

Daniel M. Yellin

Declarants further state that the above statements were made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

Date:

David M. Bashant

Date:

Donald E. Buddenbaum

Date:

8/24/2004

Michael L. Denny
Michael L. Denny

Date:

Daniel M. Yellin

Declarants further state that the above statements were made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

Date:

David M. Bashant

Date:

Donald E. Buddenbaum

Date:

Michael L. Denny

Date:

8/26/04



Daniel M. Yellin

IAFeB

Enterprise Integration Domain Version 1
Functional Specification

Created: Monday, July 17, 2000

Updated: Wednesday, August 11, 2004~~Thursday, August 03,~~
~~2000~~Monday, July 17, 20005:47 PM~~5:46 PM~~Monday, July 17, 2000

Authors: Rheanna Wellman/Paul Marr

Owners: J.G. BondMike Hallman
Don Buddenbaum

Document: EID_FunctionalSpecsification_8_02.doc
Document Version 2~~1~~0.0

PURPOSE OF THIS DOCUMENT	55
PURPOSE OF EID	66
EID FUNCTIONAL <u>SPEREQUIREMENTCS</u>	77
1 MESSAGING ARCHITECTURE	77
1.1 MESSAGE HEADER	77
1.2 CONSISTENCY WITH INSURANCE APPLICATION ARCHITECTURE (IAA)	77
1.3 ABILITY TO ADD NON IAA-XML MESSAGES	77
1.4 INETEGRATION WITH ACORD MESSAGES	77
1.5 MESSAGING TOOLS	77
2 CROSS REFERENCE FUNCTION.....	887
2.1 LDAP	88
2.2 GLOBAL UNIQUE IDENTIFIER (GUID)	88
2.3 MAINTENANCE OF THE CRF	88
2.3.1 <i>Initial Load</i>	998
3 LOGGING CAPABILITIES.....	998
3.1 MESSAGE STORAGE	998
3.2 CUSTOMIZABLE	998
4 MESSAGE MANAGEMENT	99
4.1 MESSAGING OPTIONS	10109
4.1.1 <i>'Fire and Forget'</i>	10109
4.1.2 <i>'Request/Reply'</i>	10109
4.1.3 <i>'Publish'</i>	10109
5 PROCESS MANAGEMENT (WORKFLOW)	111110
5.1 AUTOMATION OF WORKFLOW	111110
6 PARTICIPATION AND ROUTING MANAGEMENT.....	111111110
7 INTEGRATION ADAPTERS	1112111110
8 SYSTEMS MANAGEMENT	121211
9 MODEL OFFICE (IMPLEMENTED USE CASES'<u>DEFINED USE CASES'</u>)	121211
9.1.1 <i>Message Warehouse</i>	121211
9.1.2 <i>Party</i>	1313121211
9.1.3 <i>Policy / Claim</i>	1313121211
9.1.4 <i>Business Unit of Work</i>	1313121211
9.2 MODEL ADAPTERS	131312
9.3 CLIENT INFORMATION INTEGRATION SYSTEM(CIIS)	131312
APPENDIX A.....	1415141413
NON FUNCTIONAL REQUIREMENTS	1415141413
1 PLATFORMS	1415141413
1.1 NT.....	1415141413

1.2	AIX.....	1415141413
2	HARDWARE REQUIREMENTS.....	1415141413
2.1	MQSI v2.....	1415141413
3	RUNTIME PREREQUISITES	1415141413
3.1	NT PLATFORM	1415141413
3.1.1	MQ Series V 5.1 for NT.....	1415141413
3.1.2	MQSI V 2.0	1415141413
3.1.3	MQ Workflow 3.2.1.....	1415141413
3.1.4	DB2.....	1516151514
3.2	AIX PLATFORM.....	1516151514
3.2.1	MQ Series for AIX.....	1516151514
3.2.2	MQSI.....	1516151514
3.2.3	MQ Workflow.....	1516151514
3.2.4	DB2.....	1516151514
4	DEVELOPMENT ENVIRONMENT PREREQUISITES	1516151514
4.1	ALL RUNTIME PREREQUISITIES.....	1516151514
4.2	RATIONAL ROSE 2000	1516151514
4.3	XML MANAGEMENT TOOL (TBD)	1516151514
5	USABILITY.....	1516151514
6	PERFORMANCE	1516151514
1.1	PERFORMANCE WILL BE MEASURED DURING TESTING	1516151514
6.1	1516151514
6.2	SCALABILITY.....	1516151514
6.2.1	CRF Design.....	1516151514
6.2.2	Hub Design.....	1616151514
7	SECURITY.....	1617161614
8	RELIABILITY	1617161615
9	INSTALLATION.....	1617161615
9.1	AN INSTALLATION AND SETUP GUIDE	1617161615
9.2	PREREQUISITE SOFTWARE SETUP	1617161615
10	SPECIFIC DELIVERABLES TO BE PACKAGED ON CD	1617161615
10.1	DOCUMENTATION	1617161615
10.1.1	Technical Overview (Including executive overview guide)	1717161615
10.1.2	Installation and Setup Guide	1717161615
10.1.3	Customization Guide(s)	1717161615
10.1.4	Systems Management Guide.....	1718171715
1.1.1	Adapter Specification and Design Guide.....	1718171715
10.1.5	1718171715
10.1.6	Guide to Long Running Transactions.....	1718171716
10.1.7	Model Office Guide	1718171716
10.1.8	Licensing information.....	1718171716
10.1.9	Read Me file.....	1718171716
10.2	FUNCTIONAL DELIVERABLES	1718171716
1.1.1	Base EID.....	1718171716
10.2.1	1718171716
10.2.2	Model Office.....	1818181816

11 TESTING.....	<u>1819191916</u>
11.1 SYSTEM TESTING	<u>1819202017</u>
11.1.1 Entry Criteria	<u>1819202017</u>
11.1.2 Exit Criteria	<u>1819202017</u>
12 MAINTENANCE	<u>1919202017</u>
12.1 ALL DELIVERABLES WILL HAVE VERSION CONTROL	<u>1919202017</u>
12.2 ALL DELIVERABLES WILL HAVE PROBLEM MANAGEMENT CAPABILITY	<u>1920202017</u>
12.3 LEVEL 1 AND 2 SUPPORT WILL BE PROVIDED BY THE LOCAL IGS SERVICES TEAM.....	<u>1920202017</u>
12.4 LEVEL 3 SUPPORT (VIA PHONE/EMAIL) WILL BE AVAILABLE TO THE LOCAL IGS SERVICES TEAM (TERMS VIA SAN INTERNAL DOU) FROM DEVELOPMENT	<u>1920202017</u>
APPENDIX B - 'DEFINED USE CASES'	<u>2021232318</u>
<u>APPENDIX C - EXTERNAL</u>	
<u>DEPENDENCIES.....</u>	<u>20213</u>

Purpose of this Document

~~This document states the specific functional requirements for the Enterprise Integration Domain (EID) Version 1. The non functional requirements are included as an appendix.~~

The purpose of this document is to provide the functional and non-functional specifications for the EID Version 1 PRPQ product that will be satisfied by the EID V1 development plan. This document should be used in conjunction with the EID Technical Architecture document to understand the functionality of EID Version 1.

Purpose of EID

To provide an enterprise-wide scalable framework that allows multiple front-end applications (such as web and call centers) to inter-operate with back-end applications (such as policy administration and claims systems) in an effective and efficient manner.

EID Functional Specifications

1 Messaging Architecture

A common problem when integrating diverse systems is the agreement on a common set of messages and a common messages architecture. EID will use IAA-XML messages designed using version 1.3 of the IAA-XML Specification. The IAA-XML messages will be provided as a standard set related to insurance, that satisfy the requirements of the 'defined Use Cases' that are listed in the Appendix of this document.

1.1 Message Header

All IAA-XML messages used within EID will have an EID header that will hold information to improve the efficiency of the management and distribution of the messages within EID. The EID header will include an indicator with associated data tags to specify whether or not the CRF needs to be invoked, as not all message flows require CRF functionality. This will be implemented as part of the EID Header as described in the EID Technical Architecture document.

1.2 Consistency with Insurance Application Architecture (IAA)

Any insurance messages supplied with EID will be consistent with the IAA-XML standard and the IAA-XML Version 1.3 Specification. Initially use CHS V 2.2 Enhancement kit IAA-XML - IAA-XML messages EID will be able to process the IAA-XML Version 1.3 messages created to satisfy the 'defined Use Cases'.

1.3 Ability to add Non IAA-XML messages

EID will support messages that are not consistent with IAA-XML standards if they include the required EID header. ~~that are not consistent with IAA-XML standards~~ This which allows customers or business partners to add their own messages. ~~— R~~ This support for non IAA-XML messages will only include the capability to route these messages through the hub with no other message functions provided assuming that each message includes the required EID Header which will provide the routing information.

relies on functionality of MQ Series family

1.4 Integration with ACORD messages

A mapping will be provided ~~A set of adapters will to~~ allow ACORD messages to be transformed into appropriate IAA-XML messages and IAA-XML messages to be transformed into the appropriate ACORD messages. Out Of Scope - except for evaluation of the AL3 Support Pack provided by Jim MacNair to ensure that it will work within the architecture of EID. Any mapping involving either ACORD AL3 to IAA-XML or ACORD XML to IAA-XML is Out Of Scope for the EID Development Plan.

1.5 EID Message Log/Warehouse

EID will provide a Message Warehouse for storing IAA-XML messages used by EID.

1.5.1.6 Messaging tools

Tools are provided to add or change XML messages in the message warehouse. A testing tool is also provided. Documentation will be provided and the existing capabilities of MQSI V2 will be used with no special tools provided outside of MQSI V2. Provide documentation from IAA XML architect Sample IAA-XML messages will be provided for the 'defined use cases'

2 -Cross Reference Function

EID- will include a Cross Reference Function (CRF) that allows EID to maintain key key information for about party, policy and claim. It allows applications to obtain important key information about records that belong to other systems within EID. The reference data allows applications in the front end to operate independent of applications in the back end (and vice versa) by referencing the appropriate information on the database.

This will be implemented using MQSI V2 flows with nodes containing with-ESQL to connect to a CRF DB2 Table and manipulate the cross reference contents of this CRF DB2 Table . in MQSI

An entry will be provided is required for any entity that requires key translation as a result of processing an IAA-XML message

~~EID does not own any data. The system of truth is an important concept within the EID and represents within the enterprise the application that manages the integrity of the data for which it is designated as the 'system of truth'. EID will be customized at installation to indicate for each data entity (eg party) which system is designated the 'system of truth' for that entity. Messages will then be set up to use the 'system of truth' as the control point in message routing.~~

2.1 LDAP

To improve flexibility of EID there will be an LDAP directory that maintains a logical symbolic application name and the physical location of the message routing so that changes to system topology can be more easily implemented. This will be implemented with ESQL in MQSI with stored procedures in DB2 to interact with an LDAP for resolving symbolic names.

2.12.2 Global Unique Identifier (GUID)

The CRF will use a GUID as the EID internal key field. The method for generating the GUID is provided ~~that will~~ ensure that it is a unique number within the implementation of EID.

This will be implemented using with a 13-byte DB2-supplied id-creation function (DB2 V 6.1) to create the GUID.

2.22.3 Maintenance of the CRF

The CRF will need to ~~have a new entries inserted be loaded~~ modified when a new key or new application is added to EID. Configuration messages to add or change applications will ~~be included with the EID.~~

This will be implemented as part of the CRF installation/customization through manual table entries of the system identifiers and key types into defined DB2 tables in EID based on instructions provided in the EID Customization Guide.

The design of the CRF will allow it to refresh itself so that it is kept up to date when applications are changed or added to EID. This is implemented as part of the CRF installation/customization through manual table entries of the system identifiers and key types into defined DB2 tables in EID based on instructions provided in the EID Customization Guide. Automatic refresh is not planned. XML messages will be provided to create, read, update and delete data elements or records in the file. ~~By convention, this is part of implementation design~~

2.3.1 Initial Load

There is a need for an initial load to the CRF. A load utility will be provided, based on the assumption that it will be loaded from CIIS. ~~Design is to be determined~~ This will be implemented by specifying a format/structure (as documented in the EID Customization Guide) for a DB2 table to be used as an space for the data to be located before starting the CRF Load and then the Load can be initiated into CRF.

3 Logging Capabilities

As part of the overall integrity of EID-, a Business Event Log will be provided in addition to audit trail logs provided for MQ messages. This will be implemented using an ESOL warehouse subflow node in MQSI V2 with the entire contents of the IAA-XML message stored in the Event Log which will be a DB2 table.

3.1 Message storage

User specified messages are stored in a DB2 table so that key business events can be tracked, reported and analyzed. Customizations- guidance is included in the EID Customization Guide ~~can be made~~ using MQSI V2 functionality as part of EID to modify the existing EID flows to specify the specific contents of the message to be stored in a Event Log (DB2 table) as part of a services engagement. EID will initially log the entire contents of the message.

3.2 Customizable

The selection of messages that are stored is customizable, so that the user can define the types of messages that are captured. The options could include:

- Log all quotation requests
- Log all claim notifications
- Log all changes of billing address

~~This can will be customized implemented by IGS, (based on instructions in the EID Customization, using based on MOSI V2 filters as part of EID. EID will initially log all messages.~~

The model office is customized to store a ~~billing change of address~~ messages associated with the predefined business scenarios. ~~Existing MOSI V2 functions that are part of EID tools will be available for storage or messages will be used. e used~~ Documentation will be provided on how to customize the Business event log. (What about tools for analysis of the log)? (How are we delivering the Error Log)?

4 Message Management

IAA-XML Messages that are exchanged in EID -may be of different types and are managed differently depending on the message type. This allows for transactions to be handled properly, according to the business function that they are associated with. ~~##This will be implemented in EID ##using existing MOSI V2 filters. EID will provide Processes/Nodes using on MOSI V2 to process the messages and associated documentation provided in the EID Customization Guide.~~ functionality

4.1 Messaging Options

EID allows for options in defining the messages so that the right level of transaction management is applied. One of three message types is supported:

All of the following are implemented by the external application's adapter (external and separate from the EID hub) setting the appropriate values in the EID Header. ; This will be implemented in EID using MOSI V2 filters and EID Processes/Nodes to process the messages with documentation provided in the EID Customization Guide.

and using existing MOSI functionality

4.1.1 'Fire and Forget'

The message does not require a reply. MQSI also uses the terms 'datagram' and 'send and forget' for this message type. The delivery in EID- is guaranteed, based on the functionality of MQ Series.

4.1.2 'Request/Reply'

A request message is used to request a reply from another application and causes the message to wait until a reply (or time out) occurs.

4.1.3 'Publish'

~~The provider of the information (publisher) is decoupled from the consumer (subscriber) using a broker.~~ The Publish message type refers to the ability to let applications that are connected with EID know about business critical events that have occurred.

~~Applications can subscribe to those messages that are of interest to them and receive the published messages as they are generated.~~ EID relies on the functionality of MQ Series to guarantee provide this capability.

It is the responsibility of the system of truth to publish messages (via the adapter) related to add, updates, and deletes.

At customization there is a choice of the implementation of publishing messages.

A message can be published to a specific application queue and the receiving application is assured of prompt delivery of that message. This implementation requires analysis of the message and the target application and requires that as applications are added, messages are reviewed to see if the new application is required to receive the published messages. This approach is recommended where messages have a special impact on the business (e.g. high value or high risk transactions and where speed of reaction by the receiving application is important).

~~A more flexible approach, generally recommended for most messages is to publish the messages to specific queues that applications can subscribe to and receive all messages of that are contained in that queue. This is known as Publish/subscribe (pub/sub). Delivery of messages to these queues is assured but it is the responsibility of corresponding applications that require the published information to subscribe to the messages/events via their adapters, in order to receive the appropriate updates. The application may choose to receive these messages on a scheduled or periodic basis (e.g. 3 times a day) and this may not be appropriate or certain types of messages.~~

Note: Not all transactions are suitable for the publish and subscribe distributions mechanism, particularly where a reply is needed to complete a business unit or work.

Messages can flow from front end applications to backend applications (e.g. notification of billing address change) or propagate changes in the backend forward to front end applications (e.g. billing cycle updates)

5 Process Management (Workflow)

EID supports long running transactions: where the business transaction or business process needs the ability workflow to manage a series of long running or complex tasks. MQ Workflow scripts will be added for extra processing when the business scenarios are require complex actions or there is an extended transaction (e.g. consolidation of responses from different systems into a single reply) enough to need it.

EID allows the user to define the level of business process automation and process management. Some customers may require limited automation and management in the early implementation and move to increase the process content in the hub as their businesses evolve. Other customers are prepared to re-engineer their processes immediately and move the level of automation in their processes from the application or desktop to the EID. EID is flexible enough to allow a customer implementation to easily apply and change the level of process management and automation. EID uses MQ workflow to sustain a transaction and ensure that the business unit of work is completed. A sample of long running transactions will be provided as part of the model office. T

this will be implemented in EID using existing MQ Workflow processes/nodes functionality with the implementation of the 'defined Use Cases' with documentation provided in the EID Customization Guide.

5.1 Automation of Workflow

~~EID uses MQ workflow to sustain a transaction and ensure that the business unit of work is completed. Sample long running transactions are provided as part of the model office. First notice of claim scenario will be used~~

6 Participation and Routing Management

EID provides the ability to change or add participating applications and routing messages without the need to disturb the operation of current front-end and back-end applications. The application queue creation and routing tools provided with the EID allows the user to add new applications and change the routing of an existing application without disruption.

Symbolic destinations will be used in EID, which are stored in an LDAP directory.

~~When a front end application needs to route a message to a back end system (eg CHS party system), the hub will take the symbolic id of the party application, and in dispatch the message with the EID header to the appropriate queue.~~

This allows EID to react quickly to changes in application topology without having to make modifications to the application to work with EID. Only a change in the LDAP entry will be necessary. Tools are provided to maintain the LDAP directory.

This will be implemented in EID using DB2 stored procedures and ESQL functionality to access LDAP.

7 Integration Adapters

To allow messages to inter-operate with applications, integration adapters are required. No adapters will be provided with EID. Business Partners such as Siebel and ChanellePoint will

provide adapters/connectors to their applications. The EID scope is limited to the creation of an adapter design specification document and integration with Siebel 2000 Financial Services based on committed and sufficient support provided by Siebel engineers/data modellers in the timeframe of our EID development schedule. Siebel is expected to create a Siebel Connector (with support from the EID Development Team) to integrate with EID using their EAI tools to create the required Business Objects, Business Components, Integration Objects and Business Services to generate IAA-XML messages for each of the 'defined Use Cases' in the Appendix.

~~guideeeve~~ No work for EID

8 Systems Management

The IAFeB recognizes the need for consistency in systems management implementation. A common Systems Management architecture is applied to all IAFeB related implementations. EID must be capable of integrating easily with Systems Management Environment. EID can be used seswith the existing MQ Series family and Tivoli infrastructure for systems management as documented in the EID Customization Guide.

EID -will provide logging of messages, which can then be used ~~for~~ to monitor the system. ~~Im~~ This will be implemented in EID existing MOSI V2 functionality and EID will provide Processes/Nodes to process the messages and store them in the Event Log (DB2 Table) with documentation provided in the EID Customization Guide for customization.

~~plemented using~~ ESQL in MOSI

9 Model Office (Implemented Use Cases'defined Use Cases')

EID -is delivered with a model office implementation that runs on one of the designated platforms. What the model office consists of is a set of implemented ~~use-eases'~~ defined Use Cases'. A modeled set of front-end and back-end applications will be implemented with EID to show the operation of the functions of the domain. Sample data will be used. The business scenarios defined in the ~~use-eases'~~ defined Use Cases' will be included to demonstrate each of the following requirements. Appendix B contains a complete list of the Use Case information and messages. This will be delivered as EID export files from MOSI V2 and MQ Workflow, with queue manager and database manager configuration scripts and instructions along with the Use Case documentation and sample data that will demonstrate messages being processed through EID.

The installation of the Model Office wWill require customization for installation designed for production IGS-use, and does not include endpoint systems (Siebel & CIIS) as a part of the EID Model Office deliverable-packaging. CIIS currently cannot run on NT as a standalone implementation so this prevents a fully operational Model Office to be delivered on an NT-only platform. The Siebel application for the Model Office will be provided separately by Siebel along with their Connector for EID.

9.1.1 Message Warehouse

IAA-XML messages that support the Business Scenarios cases will be pre-loaded into a message warehouse. To be determined

IAA-XML Messages IAA-XML Version V1.3 messages will be created for each of the 'defined Use Cases' listed in the Appendix. The following standard messages will be included to represent Business Scenarios:
To be determined

9.1.2 Party Implemented based on available CHS V 2.2 enhancement kit IAA-XML messages

- ☐ new party
- ☐ update
- ☐ search / inquiry

9.1.3 Policy / Claim Implemented based on available CHS V 2.2 enhancement kit IAA-XML messages

- ☐ inquiry
- ☐ extended inquiry

9.1.4 Business Unit of Work To be determined

(LOB comprises Life and Financial Services, Auto, Home and P&C)

- ☐ New quotation (per LOB)
- ☐ Bind quotation (per LOB)
- ☐ Change policy (per LOB)
- ☐ First claim notification (per LOB)

9.2

9.3 Business Partner Adapters

Adapters necessary for the implementation of the Model -Office will be provided for integration with Siebel as the only external application planned for integration. EID will integrate with Siebel's ~~adapter call center~~ Financial Services application and provide the necessary functionality of the Business Scenarios. Siebel will provide an adapter/connector that handles IAA-XML messages through MQ Series.

This integration of EID with Siebel is Dependent on delivery of the Siebel adapter/connector by Siebel (with support by the EID Development Team) which will integrate Siebel to the EID via IAA-XML messages through MQ Series for the 'defined Use Cases'.

149.4 Client Information Integration System(CIIS)

CIIS is an operational client management system. In the model implementation of EID Lite, it can be thought of as a reference file that contains important summary pieces of key information about various parties in the Insurance organization. ~~The reference data is used to link together various information about a given party.~~ The data elements are based on the IAA models, and are represented by the IAA-XML messages and used in the implemented use cases 'defined Use Cases'. Integration with CIIS is dependent on the availability of the CIIS V 2.2 Enhancement Kit and the CIIS Dependencies required for EID as documented in APPENDIX C.

In a specific implementation this may be changed for an in house or other version of a party information file. This change will need the development of a new adapter to use the messages targeted at CIIS.

~~Based on the availability of the July 14th version of CIIS V 2.2 Enhancement kit, + EID dependancies as documented in the EID Technical Architecture Document., plus . Need adBased on the availa~~

Appendix A

Non Functional Requirements

1 Platforms

1.1 NT

EID -will be available on Microsoft Windows NT.

1.2 AIX

EID- will be available on AIX. However, this is dependent upon MQSI 2.0 for AIX, with a planned release of 8/31/2000.

2 Hardware Requirements

The hardware requirements for EID are based on those of the respective prerequisite products.

2.1 MQSI v2

- 500MHz processor and at least 512 MBegs of RAM.

3 Runtime PrerRequisites

3.1 NT Platform

3.1.3.1.1 MQ Series V 5.1 for NT

- Microsoft Windows NT 4.0 ~~with latest Service Pack~~
- Microsoft Internet Explorer 4.01 or higher
- Microsoft HTML Help 1.2 (refer to the directory Prereqs\HTMLHelp)
- Microsoft MMC1.1 (refer to the directory Prereqs\MMC)
- Microsoft ADSI 2.0 (refer to the directory Prereqs\ADSI)

3.1.2 MQSI V 2.0~~x.x~~

- Microsoft Windows NT 4.0 with SP 5
- Microsoft Internet Explorer V5
- MQSeries V5.1 Java Client
 - Control Center and Config Manager
 - Must be from Server CD or MA88

~~DB2~~

3.3.1.3 MQ Workflow 3.2.1

- Windows NT Version 4 and Service Pack 4
- New Windows NT User for MQWF

- ~~DB2 UDB 5.2~~
- MQSeries Server 5.1 CSD 4

~~3.43.1.4~~ DB2 6.1 V-x.x

3.2 AIX Platform (PreReq information to be provided by SWG for 8/31/00 GA)

3.2.1 MQ Series for AIX

3.2.2 MQSI

- MQSeries V5.1 Java Client
 - Control Center and Config Manager
 - Must be from Server CD or MA88

~~DB2~~

3.2.3 MQ Workflow

~~DB2 MQSeries~~

3.2.4 DB2

4 Development Environment PreRequisites

4.1 All Runtime PreRequisites

4.2 Rational Rose 2000

4.3 XML Management Tool (TBD Authority)

5 Usability

Since MQ/MQSI/MQWF are an existing IBM products, that are assumed to have gone through appropriate UCD processes, and EID doesn't require any extensive specialized programs, UCD for EID will be limited to ensuring that the documentation and packaging is done appropriately to meet UCD. Also, we plan to work with ESP customers to help ensure that UCD is addressed.

5.1 Scalability

The CRF must be able to handle up to 50 million policy references. will allow three possible configurations that support routing via a .for ,or n

6 -Performance

Performance: will be measured during testing

6.1

EID performance will be based on the underlying IBM MQ Series family and DB2 products with EID implemented in accordance to the appropriate guidelines for these products.

Throughput: ~~100,000 messages per day~~

6.2 Scalability

Scalability of EID will be based on the underlying IBM MQ Series family and DB2 products with EID implemented in accordance to the appropriate guidelines for these products to allow for scalability. Dependant on MQSI

6.2.1 CRF Design

The CRF will handle up to 50 million policy references.

6.2.2 Hub Design

The design will allow three possible configurations that support routing via a single physical hub, or a logical hub comprising multiple physical hubs, or a series of connected hubs. The impact on transaction management and workflow needs to be described based on the hub deployment / topology options.

7 Security

Security will be implemented based on existing MQ Series family capabilities — no EID specific enhancements will be made beyond the capabilities of these MQ products.

The EID needs to be able to slot into an existing enterprise wide security regime. There should be no requirement imposed by the EID to implement any particular form of security protocol. At the simplest level the EID should be a secure environment and messages that come from 'trusted' applications should be processed.

The EID should also come with more advanced security capability, so that in a 'green field' site where the EID should be able to provide (through additional SWG products) an industrial strength security regime including encryption.

2

7.8 Reliability

Reliability of EID will be based on the underlying IBM MQ Series family and DB2 products with EID implemented in accordance to the appropriate guidelines for these products to allow for reliability.

8.9 Installation

It is assumed that the systems support staff performing an installation will have the required skills in regards to all of the pre-requisite software.

9.1 An Installation and Setup guide

This guidewill provided-y guidance so that a customer, IGS or an SI partners can easily understand and execute installation of the EID Development & Runtime environment. —This

will allow the model office to be ed and customized, or for new business scenarios to be implemented.

9.2 Prerequisite Software Setup

A guide will include required configuration options for the prerequisite software.

10 Specific deliverables to be packaged on CD

9.410.1 Documentation

10.1.1 Technical Overview (Including executive overview guide)

10.1.2 Installation and Setup Guide

10.1.3 Customization Guide(s)

Guide to Defining Events to Log

10.1.3.1

Information on specifying various which messages to store in the events log will be included. Based on using MOSI

10.1.3.2

10.1.3.3 XML Messages Implementation Guide (includes IAA-XML arch guide)

10.1.3.4 Publish & Subscribe

This guide will include information on defining which types of messages may should be published. Based on using. MOSI de (includes IAA XML arch guide

X

XML Messages Implementation Guide (includes IAA XML arch guide?)

Others?

Installation and Setup Guide

10.1.4 Systems Management Guide

Adapter Specification and Design Guide

10.1.5

This document provides guidance on how to maintain a corporate standard for adapter design. It allows the enterprise to implement a standard approach to building adapters to existing or new systems reducing both initial investment and subsequent maintenance. Documentation will be provided from the Common Adapter Factory and an EID Adapter Design Specification will be provided.

User's Guide Adapter Design Guide Adapter Specification and Design Guide

10.1.6 Guide to Long Running Transactions

10.1.7 Model Office Guide

10.1.7.1 Use Cases'defined Use Cases'

10.1.8 Licensing information

10.1.9 Read Me file

10.2 FunctionalEID Deliverables

Base EID

10.2.1

Cross Reference Function dDatabase

10.2.1.1

LDAPdap Directory

10.2.1.2

10.2.1.3 Message management and testing tools

Recommendations on tools to purchase if cannot be included on CD

10.2.1.4 Message Event Log

This will be implemented using the MOSI V2. EID will use an MOSI node(s) to store all events/messages into a log/warehouse, which will be implemented as a standard DB2 table, for analysis/reporting using an external utility/program separate from EID.

10.2.2 Model Office

9.610.2.2.1 Sample Event and Error Logs

9.710.2.2.2 Siebel IAA-XML messages *Supplied by Siebel with Siebel Connector.* ~~(I don't know how to articulate this relative to the IAA-XML messages already listed?)~~

10.2.2.3 IAA-XML messages in a message warehouse *These messages will only be provided for the ~~nt~~ for the supported use-cases 'defined Use Cases' in the Appendix.*

9.810.2.2.4 A Sample Cross Reference Function dDataFile ~~(in form of DB2 tables)? Is directory a better term?~~

9.910.2.2.5 Sample Long Running Transaction using MQ Workflow ~~implemented in MQ Workflow?~~

?

Guide to Long Running Transactions

Reference Implementation Guide

Technical Overview (Including executive overview guide)

UUse Case

s

~~**Models - what is this?**~~ **IAA-XML messages in a message warehouse**

Message management and testing tools

10.2.2.6 -DB2 database for legacy host simulation ~~Ladap Directory~~

C

CIIS sample implementation ~~*Requires IGS NC*~~ ~~*Out Of Scope - nNot currently included as deliverable because it requires customization- Customization and Installation withof the CIIS database on a Ssystem*~~ ~~*390. Awaiting committed plans from CIIS Development to implement/test/verify CIIS on NT. and associated JCL*~~ ~~*cc*~~

9.1910.2.2.7 Licensing information

1011 Testing

Ability to integrate with multiple 'front end' applications, multiple data warehouses and multiple backend systems but tested with:

1. Siebel 2000 V x.x

2. S

- with:
- Siebel 2000
- Siebel 2000 Enterprise Application Integration V x.x
- 3. CIIS V x.x
- 4. Test DB2 database to simulate Legacy system V 1.0

10.111.1 System Testing

10.1.1A System Test plan will be created and approved prior to system test start

10.1.211.1.1 Entry Criteria

A least 1 Use Case working end-to-end "Siebel, CIIS, Legacy test DB"

10.1.311.1.2 Exit Criteria

10.1.3.111.1.2.1 All 17 Use Cases 'defined Use Cases' working end-to-end

- All Sev 1 and Sev 2 problems resolved
- All Sev 3 problems have an agreed-to plan to resolve
- All Sev 4 problems have an agreed-to plan to resolve prior to next release

40.2

1112 Maintenance

11.112.1 All deliverables will have version control

11.212.2 All deliverables will have problem management capability

11.312.3 Level 1 and 2 support will be provided by the local IGS services team

11.412.4 Level 3 support (via phone/email) will be available to the local IGS services team (terms via an internal DOU) from development

12 Documentation

A Developer's Guide to EID will be provided. It will contain the necessary information for integrating EID into a customer solution. It will specifically contain those guides that are listed here.

12.1 Systems Management

A guide to EID Systems Management Consideration.

12.2 Installation

A complete installation guide, which describes pre-configured options and choices made during development and the customization options and installation tasks required.

12.3 Adapter Specification

12.3.1 A guide will be provided detailing the requirements for an adapter to interface with EID Design

This document provides guidance on how to maintain a corporate standard for adapter design. It allows the enterprise to implement a standard approach to building adapters to existing or new systems reducing both initial investment and subsequent maintenance.

12.3.2 Create

Creates adapters to support business activity. It provides guidance a standard approach that builds adapters quickly and reduces maintenance overhead.

12.4 Customization Guides

A set of customization guides and tools that allow the user to define the messages that get stored in the event log.

Ex: Log all quotation requests, Log all claim notifications, Log all changes of billing address

12.5 Messages

12.5.1 Architecture

This document provides a corporate standard for XML messages. The benefit is that developers have a standard approach to messages and messaging.

12.5.2 Message Data Warehouse User Guide

A set of instructions will be provided for adding new XML messages to the Message Data Warehouse.

12.5.2.1Create Message

Procedure supports the business activity and provides reuse of messages throughout the enterprise. Any business in the enterprise can use this process to define the specific content and shape of their own message variant.

12.5.3Messaging types

12.5.3.1Publish and Subscribe

EID comes with a user guide for managing the Publish and Subscribe function. It includes instructions for defining which types of messages need to be published to other applications.

12.6Cross Reference Data

12.6.1Create Routing

Creates routing tables and allows the business to define routings within the installation routing framework.

12.6.2Create Participants

Creates a database of systems and applications. Typically an infrastructure function that allows the infrastructure owner to define which applications and which systems are available to the integration layer.

Appendix B – 'defined Use Cases'

Priority	Use Case Number	Use Case Description	Scenario	Scenario Source	Line of Business
1	NA0003	Addition of Customer Name, Address, and Personal Data			All
2	NA0002	Change Address for Name and Address Record	Customer Service (Change of Address) - Scenario 3; Update Policy Details	Dallas COI; Model Business Scenarios	All
3	NA0001	Inquiry into an Existing Address Record			All
4	CM0001	First Notice of a Life Death Claim	Add a New Claim	Model Business Scenarios	Life
4	CM0002	First Notice of Claim on Auto Policy	Add a New Claim	Model Business Scenarios	Auto
4	CM0003	First Notice of Claim on Homeowners Policy	Add a New Claim	Model Business Scenarios	P&C
5	CM0004	Inquiry into Status of Death Claim	Claim Inquiry	Model Business Scenarios	Life
5	CM0005	Inquiry into Status of Homeowners Claim	Claim Inquiry	Model Business Scenarios	P&C
5	CM0006	Inquiry into Status of Auto Claim	Claim Inquiry	Model Business Scenarios	Auto
5	PS0004	Inquiry into Auto Coverage Billing Account	Policy Inquiry	Model Business Scenarios	P&C
6	NB0002	Addition of Policy Application for Traditional Life Policy	Client Responds to Campaign - Scenario 2; Add a New Policy	Dallas COI; Model Business Scenarios	Life
6	NB0007	Addition of auto policy	Add A New Policy	Model Business Scenarios	Auto

6	NB0008	Addition of homeowners policy	Add a New Policy	Model Business Scenarios	P&C
7	PS0005	Quote for Addition of Personal Articles Coverage for Existing Homeowners Policy	Policy Inquiry	Model Business Scenarios	Auto
8	PS0002	Request to Process Partial Surrender Quote	Customer Service on the Internet - Scenario - 5; Policy Inquiry	Dallas COI; Model Business Scenarios	Life
9	HS0001	Inquiry into history of events - history as part of each use case	User Event Log	Model Business Scenarios	All
10	PR0003	Campaign Generation	Customer Service - Scenario 1	Dallas COI	Life

The following standard messages will be included to represent Business Scenarios:

Party *This will be implemented only as required for the 'defined Use Cases' and based on available IAA-XML V1.3 messages supported by the CIIS V 2.2 Enhancement Kit.*

- new party
- update
- search / inquiry

Policy / Claim *This will be implemented only as required for the 'defined Use Cases' and based on available IAA-XML V1.3 messages supported by the CIIS V 2.2 Enhancement Kit*

Inquiry *This will be implemented only as required for the 'defined Use Cases' and based on available IAA-XML V1.3 messages supported by the CIIS V 2.2 Enhancement Kit*

- extended inquiry

Business Unit of Work *This will be implemented only as required for the 'defined Use Cases' and based on available IAA-XML V1.3 messages supported by the CIIS V 2.2 Enhancement Kit*

(LOB comprises Life and Financial Services, Auto, Home and P&C)

- New quotation (per LOB)
- Bind quotation (per LOB)
- Change policy (per LOB)

- First claim notification (per LOB)

Appendix C – External Dependencies

Implementation of EID depends on the following items that are out of scope of EID architectural control.

CIIS Party System

A working EID requires systems at the end points for integration. A party system is a fundamental component of any enterprise. EID is designed presuming that the party system of truth contains at a minimum, the functionality available in the CIIS system version. EID requires the following extensions to CIIS:

- CIIS must be capable of notifying EID that a CRUD¹ level event has taken place for the information under CIIS control². EID provides the infrastructure for synchronization of the entire enterprise, but CIIS must provide the event trigger in the form of IAA-XML messages on the appropriate MQ queues.
- CIIS must provide an API for user authentication. The API will take a user id and password and return a client id³ on a successful hit.
- Provide access to CIIS information via IAA-XML edition 5 messages, including:
 1. CRUD Party
 2. CRUD Policy Summary
 3. CRUD Claim Summary
- Scalability to meet EID non-functional requirements.
- Data model harmonization with other EID applications such as ChannelPoint.

IAA-XML

¹ Create, Read, Update, and Delete activities.

² Control is based on being the system of truth for a specific entity. In the case of CIIS, this is party information.

³ The client id will have been adjusted to meet

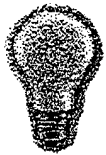
EID uses the EID header to drive all EID functionality, and so IAA-XML message generation is beyond the scope of the EID hub. Instead, EID has the following requirements for IAA-XML in order to deliver on the demonstration requirements for the EID

- Accord (XML and AL3) interoperability for IAA-XML (mapping).
- IAA-XML messages for the 'defined Use Cases' defined in the EID functional specifications, supporting CIIS and Siebel.
- IAA-XML support for the EID message header.
- IAA-XML versioning, including a released version matching the EID delivery.

Siebel.

There is a significant dependency on Siebel for interoperation. These commitments will be documented and signed-off by IBM IAFEB management team and Siebel management in the Siebel-IBM Joint Development Agreement and development plans will be interlocked between Siebel and IBM to ensure that the appropriate dependencies are met to meet the EID Development Plan/Schedule.

EXHIBIT B



Disclosure SMS8-2000-0059

Created By: Daniel Yellin Created On: 08/14/2000 01:40:12 PM

Last Modified By: Daniel Yellin Last Modified On: 09/22/2000 12:20:41 PM

*** IBM Confidential ***

Required fields are marked with the asterisk (*) and must be filled in to complete the form.

Summary

Status	Under Evaluation
Original Location	SOM, SMS
Processing Location	END
Functional Area	GS-(HEALD) IBM Global Services
Attorney/Patent Professional	John Pivnichny/Endicott/IBM
IDT Team	Richard Malek/Endicott/IBM; Robert L King/Endicott/IBM; Jon B. Martens/Endicott/IBM; Fred Rogers/Endicott/IBM
Submitted Date	08/28/2000 12:12:36 AM
Owning Division Select	GS
PVT Score Calculate	To calculate a PVT score, use the 'Calculate PVT' button.
Lab	HEALD
Technology Code	

Inventors with Lotus Notes IDs

Inventors: Daniel Yellin/Watson/IBM, Donald Buddenbaum/Indianapolis/IBM@IBMUS, Michael Denny/Charlotte/IBM, David Bashant/Charlotte/IBM

Inventor Name

> denotes primary contact

Inventor Name	Inventor Serial	Div/Dept	Manager Serial	Manager Name
> Yellin, Daniel M.	221701	7J/NK6A	113055	King, Kenneth S. (Ken)
Buddenbaum, Donald E.	825182	7J/NK6A	221701	Yellin, Daniel M.
Denny, Michael L.	804059	07/LR8D	303641	Brown, C.C. (Carol)
Bashant, David M.	894469	07/LR8B	319512	Casey, Sylvia A.

Inventors without Lotus Notes IDs

IDT Selection

Main Idea

*Title of disclosure (in English)

A mechanism for dynamically tracking related data elements in disparate systems and its facilitation in enterprise system integration

*Idea of disclosure

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

Companies have many systems with data replicated between these systems. (Example: an insurance company may maintain the name and address and related information regarding its customers in a central client application, as well as in different product systems, as well as in billing and claims systems.)

As new data is created, modified, or deleted in one system, these changes often have to be propagated to other systems, so that the data is kept consistent ("synchronized"). Many techniques are used today to help solve this problem. One such mechanism is "publish and subscribe". Even using these techniques, it still requires effort on the part of the application, upon receiving notification of a change made to specific data in another application, or upon a request to act upon data that corresponds to data in another application, to figure out where this data resides in its own system. (For instance, if one application changes the information related to a specific customer, and another application is informed of this change, the application informed of the change must somehow figure out where the changed data resides in its system in order to incorporate the changes to its system.)

Without loss of generality, we will assume that each system stores information in records, and each record has an associated Identifier (ID). A record may correspond to a table row in a relational data base, an object in an object-oriented system, or some other unit of data storage. In any case, we assume that each record stores some set of related information, and we call each "atomic" piece of data stored in a record a data property. Each ID associated with a record in a system uniquely identifies that record in that system. That means, given an system specific ID, the system can find and retrieve the associated record.

To solve this problem stated above, some systems have used a Cross Reference Facility (xRef facility). The xRef facility lists, for each collection of information, the record IDs in each system associated with this data collection. For instance, for each customer, there will be a ID for each system. This ID will be the ID of the record where this customer's information can be found in that particular system. When one system changes a particular customer record, other systems will be informed of the change. Each system informed of the change will be sent its own record ID along with the information being changed, so it can easily find and update the customer information being changed in its own system. This solution works if there is a one-to-one correspondance between records in each system -- if each system has exactly one ID associated with the same collection of information -- with the same set of properties. This is often not the case. Rather, one system may store all the data properties in one record, another system may store some of the data properties in one record and the rest of the properties in another record, and yet a third system may store the properties in five different records. Furthermore, these solutions often require much work on the part of the application to maintain an up-to-date xRef facility.

This invention facilitates the ability to maintain consistency of replicated information across multiple systems even when the way these systems store the data is very different. It enhances and improves on the xRef facility approach. There are several advantages to the mechanism described in this invention: First, it allows multiple systems to have different record structures for storing related information, but still be able to easily find the information and make changes to the data in response to changes made by another system. Second, it automates much of the process of maintaining correspondance between the IDs in multiple systems, thereby mitigating the amount of work each application must do on its own. Overall, this mechanism makes it much easier to maintain consistency between replicated data stored in multiple systems.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

1. Overview: The integration hub

The technique described in this patent presumes a central facility, called an Integration Hub. This facility communicates with all the different systems, and presumes that all communication regarding changes in data is passed between these systems and the hub. This avoids systems from having to know about other systems, simplifies communication patterns, and centralizes the administration and implementation of business policies. The notion of an Integration Hub is not new; this novelty described in this patent is how the Integration Hub maintains correspondance between Identifiers in various systems. This increases the value of the Integration Hub by shifting more of the burden to the Integration Hub and

requires less work by each individual system. This in turn makes it easier and cheaper to integrate systems.

In particular, when one system creates, updates, deletes, or requests to read some data, and this information needs to be transferred to other systems, the system initiating this action sends the request to the hub (in a format described below). The sending system need not be aware of which systems will receive this request, as this is handled by the hub. The sending system need not be aware of any Identifiers needed by the receiving system(s) to act upon the request. This will be automatically supplied by the hub as it forwards the request to the intended system(s). Similarly, the receiving system(s) need not be aware of the requesting system, or of any Identifiers needed by the sending system to process the reply (when a reply is required). The receiving system(s) will process the request and return its reply to the hub. The hub will forward the request back to the requesting system, providing the Identifier(s) required by the sending system. The management of these Identifiers is maintained even as new data is created, existing data is replicated by additional systems, or existing data is deleted from existing systems. The management of these Identifiers is maintained even if the number of Identifiers required for a collection of data properties is different for different systems, and even if the way these Identifiers is allocated to the data properties is different for different systems. This is an important innovation as many systems have very different ways of storing their data, and managing the relationship between these Identifiers is a tedious and error-prone process. Automating this process within the Integration Hub makes it much easier to integrate systems.

II. Data structures used

The following data structures and message formats are used by the technique described in this patent:

1. Data Model of data being replicated across multiple systems

To employ this technique, one first creates a data model for the data being replicated amongst the various systems, or uses an existing data model. This data model is ultimately expressed in the data markup language XML (Extensible Markup Language). This data model consists of a collection of data properties. Each data property has a Property Type Name associated with it (e.g., Integer, Character, Reference To Person, ...). Data properties are partitioned into a set of data aggregates, so that each property belongs to one and only one data aggregate. Each aggregate has an associated Aggregate Type Name (e.g., Person, Organization, Financial Account, ...).

[One valid type for a data property is a reference to a data aggregate. Such a data property would have the Property Type Name "Reference to DataAggregateType", where "DataAggregateType" is an Aggregate Type Name. Because the data model is expressed in XML, the representation is self describing. That is, the representation in XML explicitly states the Aggregate and Property Type Names of all data elements.]

The actual data model used is not specified by this patent, and can be applied to any data model that conforms to the above characteristics. It may be an accepted standard data model or one specific for this integration hub and the systems it supports.

2. A Cross Reference Table (xRef table).

Each entry in xRef Table corresponds to a data aggregate being replicated across multiple systems. For each such data aggregate, the entry in the xRef table consists of the following:

- i) a Globally Unique Identifier (GUID) associated with this aggregate;
- ii) the Aggregate Type Name associated with this aggregate;
- iii) a boolean field, IsActive. If IsActive has the value "True", then the aggregate is known to be currently active (stored) in at least one system. If this value is "False", then the aggregate is not known to be stored in any system;
- iv) for each system S in which this aggregate is known to be stored, the following additional information is

stored: the record IDs of the records in which this aggregate is stored in S, the name of the system S, and an optional "cookie", which contains system specific information related to this aggregate. (As discussed below, this cookie may contain, for example, additional information for the system to use to facilitate mapping between the record ID and its internal storage). Note that this table allows one to associate with any single aggregate, for any system S, a set of Record IDs for this aggregate. This may be a single Record ID if S stores the data properties of the aggregate in a single record structure, or multiple record IDs if it stores these properties in multiple record structures. Note also that it is possible for a single Record ID for a system S to be associated with multiple aggregates. This would be the case, for instance, if a single record in system S stored properties associated with multiple aggregates of the data model. However, we place the following restriction: for any system S, there exists only one aggregate in which S associates the particular set of system ID(s) Id-1, Id-2, ..., Id-n. That is, system S may associate an particular ID with multiple aggregates as part of "key" for the aggregate. But each set of system IDs for any given aggregate must be unique in order to enable the hub to uniquely identify the aggregate given a particular set of system specific IDs. However, see below for cases where it is possible to relax this rule.

~ es :h
none
Address
Ac System
in S
Assess

We assume that a particular entry in xRef table can be found in one of two ways: First, given a GUID, the unique aggregate associated with this GUID can be retrieved from the table. Second, given a system name and a set of system specific IDs associated with a particular aggregate in this system, the unique aggregate associated with these IDs can be retrieved from the table.

There exists one logical xRef table in the integration hub, although it may be partitioned into separate physical implementations for various reasons such as performance. A graphical view of the xRef table is given in figure 1:

GUID	Aggregate Type Name	IsActive
Record IDs for System1	System1 Name	Cookie1
Record IDs for System2	System2 Name	Cookie2
...

3. Aggregate Keys

A key data structure associated with an aggregate A is represented by four XML elements, a GUID element, an AGGREGATE TYPE NAME element, a SYSTEM DATA element, and an ACTION CODE element. The SYSTEM DATA element contains a System Name, a list of Record IDs, and optionally a system specific Cookie. The XML structure of an Aggregate Key is given in figure 2:

```
<KEY>
  <GUID>
  <\GUID>
  <AGGREGATE TYPE NAME>
  <\AGGREGATE TYPE NAME>
  <SYSTEM DATA>
    <SYSTEM NAME>
    <\SYSTEM NAME>
    <IDENTIFIERS>
      <ID>
      <\ID>
      <ID>
      <\ID>
      ...
    <\IDENTIFIERS>
    <COOKIE>
    <\COOKIE>
```



```

<\SYSTEM DATA>
<ACTION CODE>
<\ACTION CODE>
<KEY>

```

We assume that everytime an aggregate is sent in a message, this key structure is sent with it, although some elements of the structure may or may not be present in any particular message, as described below. The <GUID> is a globally unique ID created by the hub to uniquely identify this aggregate. The <Aggregate Type Name> is the type name of this aggregate, as specified by the data model. The <SYSTEM NAME> element and the <IDENTIFIERS> element contain the internal system IDs associated with this aggregate in the particular system known by the given system name. The <COOKIE> element is an uninterpreted string of information that a particular system wants to store with this <GUID>. The <ACTION CODE> can have the following values: "Created", "Exists", "Deleted", "AttachID". The use of these ACTION CODEs in a message are described below.

III. Mapping system specific IDs to aggregate GUIDs

Describe here the various cases of one-to-one, one-to-many, many-to-one, and many-to-many mappings between system records and aggregates.

IV. Algorithms for maintaining the xRef table and for substituting the appropriate IDs as messages are sent to and from the integration hub

Every message sent through the hub operates on aggregates. The semantics of each message determines whether any particular aggregate in the message is a newly created aggregate (has never been seen before by the hub, and therefore did not have a GUID associated with it prior to this message invocation), is an existing aggregate (which the hub has already created a GUID for) in which the message is simply "operating on" (e.g., may be changing semantic content of the data properties, or may be asking to read the data associated with this aggregate in another system, or may be required for an arbitrary business action as determined by the semantics of the message), or is an existing aggregate (for which the hub has already created a GUID) being deleted from a particular system. Finally, a message may be sent to the hub by a system asking it to associate its system specific ID(s) with an existing aggregate or to update the system specific ID(s) it has already associated with a specific aggregate. Hence we describe the actions performed by the sending system, by the hub, and by the receiving system, for each of these cases. Since any message may have multiple aggregates, this describes the actions taken by the systems and hub for every aggregate in a message.

For any message the hub receives, the semantics of the message will usually dictate that the message is intended for one or multiple other systems. The intended recipient(s) of the message may be explicitly stated in the header of the message, or it may be inferred by the hub. For instance, the hub may utilize a publish and subscribe algorithm, in which the sender sends the message to the hub, and the hub forwards the message to all systems that have previously indicated interested in receiving messages of this nature. We assume that the hub utilizes some mechanism for determining which systems it is to send (forward) messages it receives, but the exact mechanism used is independent of mechanism described in this patent.

Case 1: The sending system is creating a new aggregate.

In this case, the sending system would leave the GUID element empty, but include the AGGREGATE TYPE NAME and any system IDENTIFIERS it wants to associate with this aggregate. It must associate at least one IDENTIFIER with this aggregate. It would include its SYSTEM NAME, and optionally add any private information it wants to associate with this aggregate in the COOKIE element. The ACTION CODE for this aggregate would be set to "Created".

The hub, upon receiving this message, would create a new GUID for this aggregate, and create a new entry in xRef table. This entry would have this GUID in its GUID field, and its IsActive field would be set

to TRUE. It would take the SYSTEM NAME, IDENTIFIERS, and COOKIE contained in the KEY structure and enter these into this xRef table entry. Finally, the hub would forward this message to the intended recipient(s) of the message. When it forwards the message, it keeps the KEY structure the same as how it was received, with the following exceptions: it deletes the SYSTEM IDENTIFIERS (so that the IDENTIFIERS element becomes EMPTY), sets the COOKIE and SYSTEM NAME elements to be empty, and sets the GUID to be the value of the newly created GUID.

Each receiving system decides whether or not it wants to locally store a copy of this aggregate. If so, it will create system record(s) containing the data properties associated with this aggregate. It can either internally associate the GUID with these system record(s), or it can use its own system specific ID(s) for this record. In the latter case, it needs to inform the hub of the value of these ID(s) to be used in future messages. It does so by sending a special AttachID message to the hub. This is described in case 4.

Case 2: The sending system is operating on an existing aggregate

In this case, the sending system either knows the GUID associated with the aggregate it is operating on, or has system specific ID(s) associated with this aggregate. In the former case, it would set the <GUID> element to the GUID for this aggregate and leave the <IDENTIFIERS> element empty. In the latter case, it would leave the GUID element empty but send its system specific ID(s) in the <IDENTIFIERS> element. Since the hub already knows the type of this aggregate, the sending system may optionally leave the <AGGREGATE TYPE NAME> element empty. The <COOKIE> element should be left empty, and the <ACTION CODE> must be set to "Exists".

The hub, upon receiving this message, would determine which systems to forward this message to. Since this aggregate already exists, there is an entry for this aggregate in the xRef table. If the sending system sent the GUID for this aggregate, the hub can retrieve the xRef entry E for this aggregate by using this unique GUID. Otherwise, the sending system sent system specific IDs for this aggregate. As dictated above, this means that these IDs uniquely identify the aggregate, and the hub can use these IDs to retrieve the xRef entry E for the aggregate associated with these IDs.

For each system S to which the hub needs to forward this message, S may or may not already have an entry associated with this aggregate in the xRef entry E. First consider those systems S which already have an entry associated with E. In this case, the hub looks up the entry for this system associated with E, retrieving the Record ID(s) and Cookie (if present) for this system. The hub forwards this message to S, sending the KEY structure with the GUID and AGGREGATE TYPE NAME set according to the entry in the xRef table. The SYSTEM NAME, IDENTIFIERS, and COOKIE would be set according to the information given in entry E. The ACTION CODE would be set to "Exists". When S receives this message, because the action code is "Exists" and because its system specific ID(s) are included in the KEY structure, it can find those system record(s) associated with this aggregate, and operate on the record(s) as dictated by the semantics of the message. If needed, it may use the COOKIE to facilitate its ability to locate the records associated with this aggregate.

Second consider those systems S which the hub needs to forward this message to but do not have an entry associated with E. In this case, the hub forwards the message to S, sending the KEY structure with the GUID and AGGREGATE TYPE NAME set according to the entry in the xRef table. The SYSTEM NAME, is set to the system being sent to, but the IDENTIFIERS and COOKIE elements are set to be empty. The ACTION CODE would be set to "Exists". When S receives this message, because the action code is "Exists" and because no system specific ID(s) are included in the KEY structure, it knows that the aggregate being operated upon has been created previously in other systems but it has no associated SYSTEM IDs associated with this aggregate. The system may be able to use the GUID and AGGREGATE TYPE NAME to identify an existing system record(s) corresponding to this aggregate, may want to create new system specific record(s) for this aggregate, or may take some other action. What this specific system does upon receiving this message is system specific and is not dictated by the mechanism of this patent. However, if the system decides after receiving this message to associate system specific records with this aggregate, and to use system specific ID(s) for these records, it needs

to inform the hub of the value of the system specific record ID(s) to be used in future messages regarding this aggregate. It does so by sending a special AttachID message to the hub. This is described in case 4.

Case 3: The sending system is deleting an existing aggregate from its internal system

In this case, the sending system either knows the GUID associated with the aggregate it is operating on, or has system specific ID(s) associated with this aggregate. In the former case, it would set the <GUID> element to the GUID for this aggregate and leave the <IDENTIFIERS> element empty. In the latter case, it would leave the GUID element empty but send its system specific ID(s) in the <IDENTIFIERS> element. Since the hub already knows the type of this aggregate, the sending system may optionally leave the <AGGREGATE TYPE NAME> element empty. The <COOKIE> element should be left empty, and the <ACTION CODE> must be set to "Deleted".

The hub, upon receiving this message, would determine which systems to forward this message to. Since this aggregate already exists, there is an entry for this aggregate in the xRef table. If the sending system sent the GUID for this aggregate, the hub can retrieve the xRef entry E for this aggregate by using this unique GUID. Otherwise, the sending system sent system specific IDs for this aggregate. As dictated above, this means that these IDs uniquely identify the aggregate, and the hub can use these IDs to retrieve the xRef entry E for the aggregate associated with these IDs. Since the sending system S has deleted this aggregate from its system, the hub can remove from E the entry associated with S (if any). If after removing this entry, there is no longer any system records associated with E, then the hub sets the IsActive field to "FALSE". Once an entry in the xRef table has its IsActive field set to false, it is subject to removal. Whether or not it gets removed from the table, and when this removal occurs, depends upon table management and archival processes.

The hub would then forward the message to each intended recipient system S, setting the KEY structure in the same manner as done above in case 2. The only exception is that the ACTION CODE element would be set to "Deleted".

Case 4: A system wants to associate its internal IDs with an existing aggregate, or to update its internal IDs already associated with an aggregate

A system, usually -- but not exclusively -- at the point in time of learning of the creation of a new aggregate with a given GUID, can create a replicated version of the aggregate and tell the hub to associate its system record ID(s) with this aggregate. It may do so in a message responding to the message in which it was informed about this new aggregate, or may do so in a special message intended only for the hub. In either case, it would create a KEY structure as follows: it sets the GUID element to the GUID of the aggregate, and may optionally set the AGGREGATE TYPE NAME element. It sets the SYSTEM NAME element to its own system name, sets the IDENTIFIER element to contain the ID(s) of its records corresponding to this aggregate, and optionally sets the COOKIE element to contain any private information it wants associated with this aggregate. It sets the ACTION CODE to "AttachID". Upon receiving this KEY structure, the hub inserts a new row into the xRef table entry associated with this aggregate (which it identifies by the GUID element in the KEY structure) by putting the SYSTEM NAME, IDENTIFIERS, and COOKIE contained in the KEY structure into the appropriate fields of this new row.

Once a system has requested that the hub associate system specific ID(s) with a particular aggregate, it may later wish to change the ID(s) associated with that aggregate. This may be done, for instance, to reorganization of the system specific data within the host system. In this case the system would send a message to the hub containing a modified KEY structure. This KEY structure would be identical to the one described in section II above, with the following exception: instead of an IDENTIFIERS element, it would contain an <OLD IDENTIFIERS> element and a <NEW IDENTIFIERS> element. Each of these elements would themselves contain a repeating <ID> element (i.e., one or more ID(s)). The sending system would optionally leave the GUID and AGGREGATE TYPE NAME elements empty, would set its

name to the SYSTEM NAME element, set the <OLD IDENTIFIERS> element to contain the ID(s) it previously associated with this aggregate, and set the <NEW IDENTIFIERS> element to contain the new ID(s) it wants associated with this aggregate. It would optionally set the COOKIE element to contain any private information it wants to associate with this aggregate. The ACTION CODE for this aggregate would be set to "AttachID". Upon receiving this KEY structure, the hub first retrieves the xRef entry E associated with this aggregate, which it would find using either the GUID, if supplied, or the system ID(s) contained in the <OLD SYSTEM IDENTIFIERS> element. It would then replace the row in E allocated to this system with the a new row, by putting the SYSTEM NAME, NEW IDENTIFIERS, and COOKIE contained in the KEY structure into the appropriate fields of this new row.

V. Enhancements for supporting creation requests

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?

Need to check more thoroughly offerings by other system integrator software vendors. In particular, CrossWorlds may even have some patents in this area, but we believe that our approach is different.

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.

It will be released in a product within a couple of months.

*Critical Questions (Questions 1 - 7 must be answered)

*Question 1	
On what date was the invention workable? 06/01/2000 Please format the date as MM/DD/YYYY (Workable means i.e. when you know that your design will solve the problem)	

*Question 2		<input checked="" type="radio"/> Yes
Is there any planned or actual publication or disclosure of your invention to anyone outside IBM?		<input type="radio"/> No
If yes, Enter the name of each publication or patent and the date published below.		
Publication/Patent: TBD		
Date Published or Issued: Not yet submitted		
Are you aware of any publications, products or patents that relate to this invention?		<input type="radio"/> Yes
		<input checked="" type="radio"/> No
If yes, Enter the name of each publication or patent and the date published below.		
Publication/Patent:		
Date Published or Issued:		

*Question 3		<input checked="" type="radio"/> Yes
Has the subject matter of the invention or a product incorporating the invention been sold, used internally in manufacturing, announced for sale, or included in a proposal?		<input type="radio"/> No
Is a sale, use in manufacturing, product announcement, or proposal planned?		<input checked="" type="radio"/> Yes
		<input type="radio"/> No
If Yes, identify the product if known and indicate the date or planned date of sale, announcements, or proposal and to whom the sale, announcement or proposal has been or will be made.		
Product: Not yet named, possibly Websphere Integrator, Insurance Edition		
Version/Release: First version		
Code Name: Insurance Application Framework for e-Business, Enterprise Integration Domain		
Date: End of September/October 2000		
To Whom: SWG product		
If more than one, use cut and paste and append as necessary in the field provided.		

Question 4 Was the subject matter of your invention or a product incorporating your invention used in public, e.g., outside IBM or in the presence of non-IBMers?	<input type="radio"/> Yes <input checked="" type="radio"/> No
If yes, give a date. Please format the date as MM/DD/YYYY	

Question 5 Have you ever discussed your invention with others not employed at IBM?	<input type="radio"/> Yes <input checked="" type="radio"/> No
If yes, identify individuals and date discussed. Fill in the text area with the following information; the names of the individuals, the employer, date discussed, under CDA, and CDA #.	

Question 6 Was the invention, in any way, started or developed under a government contract or project?	<input type="radio"/> Yes <input checked="" type="radio"/> No <input type="radio"/> Not sure
If Yes, enter the contract number	

Question 7 Was the invention made in the course of any alliance, joint development or other contract activities?	<input type="radio"/> Yes <input checked="" type="radio"/> No <input type="radio"/> Not Sure
If Yes, enter the following :Name of Alliance, Contractor or Joint Developer	
Contract ID number	
Relationship contact name	
Relationship contact E-mail	
Relationship contact phone	

Question 8 Have you submitted, or are you aware of, any related disclosure submission?	<input checked="" type="radio"/> Yes <input type="radio"/> No
If Yes, please provide the title and docket or disclosure number below:	
I believe that the company CrossWorlds has products that address this issue and may have patents on their technology for this.	

Question 9 What type of companies do you expect to compete with inventions of this type? <i>Check all that apply.</i>
<input checked="" type="checkbox"/> Manufacturers of enterprise servers <input type="checkbox"/> Manufacturers of entry servers <input type="checkbox"/> Manufacturers of workstations <input type="checkbox"/> Manufacturers of PC's <input type="checkbox"/> Non-computer manufacturers <input type="checkbox"/> Developers of operating systems <input checked="" type="checkbox"/> Developers of networking software <input checked="" type="checkbox"/> Developers of application software <input checked="" type="checkbox"/> Integrated solution providers <input checked="" type="checkbox"/> Service providers <input type="checkbox"/> Other (Please specify below)

Patent Value Tool (Optional - this may be used by the inventor and attorney to assist with the evalu

(The Patent Value tool can be used by you or the evaluation team to determine the potential licensing value of your invention.)

The **Patent Value Tool** has not yet been used to calculate a score.

Post Disclosure Text & Drawings

Enter any additional information relating to this disclosure below:

9/13/00

Here is an update to section 2 above. It shows the changes by text strikeouts and highlighting (in red text). The main change is to the end.

I. Overview: The integration hub

The technique described in this patent presumes a central facility, called an Integration Hub. This facility communicates with all the different systems, and presumes that all communication regarding changes in data is passed between these systems and the hub. This avoids systems from having to know about other systems, simplifies communication patterns, and centralizes the administration and implementation of business policies. The notion of an Integration Hub is not new; this novelty described in this patent is how the Integration Hub maintains correspondance between Identifiers in various systems. This increases the value of the Integration Hub by shifting more of the burden to the Integration Hub and requires less work by each individual system. This in turn makes it easier and cheaper to integrate systems.

In particular, when one system creates, updates, deletes, or requests to read some data, and this information needs to be transferred to other systems, the system initiating this action sends the request to the hub (in a format described below). The sending system need not be aware of which systems will receive this request, as this is handled by the hub. The sending system need not be aware of any Identifiers needed by the receiving system(s) to act upon the request. This will be automatically supplied by the hub as it forwards the request to the intended system(s). Similarly, the receiving system(s) need not be aware of the requesting system, or of any Identifiers needed by the sending system to process the reply (when a reply is required). The receiving system(s) will process the request and return its reply to the hub. The hub will forward the request back to the requesting system, providing the Identifier(s) required by the sending system. The management of these Identifiers is maintained even as new data is created, existing data is replicated by additional systems, or existing data is deleted from existing systems. The management of these Identifiers is maintained even if the number of Identifiers required for a collection of data properties is different for different systems, and even if the way these Identifiers is allocated to the data properties is different for different systems. This is an important innovation as many systems have very different ways of storing their data, and managing the relationship between these Identifiers is a tedious and error-prone process. Automating this process within the Integration Hub makes it much easier to integrate systems.

II. Data structures used

The following data structures and message formats are used by the technique described in this patent:

1. Data Model of data being replicated across multiple systems

To employ this technique, one first creates a data model for the data being replicated amongst the

various systems, or uses an existing data model. This data model is ultimately expressed in the data markup language XML (Extensible Markup Language). This data model consists of a collection of data properties. Each data property has a Property Type Name associated with it (e.g., Integer, Character, Reference To Person, ...). Data properties are partitioned into a set of data aggregates, so that each property belongs to one and only one data aggregate. Each aggregate has an associated Aggregate Type Name (e.g., Person, Organization, Financial Account, ...).

~~{One valid type for a data property is a reference to a data aggregate. Such a data property would have the Property Type Name "Reference to DataAggregateType", where "DataAggregateType" is an Aggregate Type Name. Because the data model is expressed in XML, the representation is self-describing. That is, the representation in XML explicitly states the Aggregate and Property Type Names of all data elements.}~~

The actual data model used is not specified by this patent, and can be applied to any data model that conforms to the above characteristics. It may be an accepted standard data model or one specific for this integration hub and the systems it supports.

2. A Cross Reference Table (xRef table).

Each entry in xRef Table corresponds to a data aggregate being replicated across multiple systems. For each such data aggregate, the entry in the xRef table consists of the following:

- i) a Globally Unique Identifier (GUID) associated with this aggregate;
- ii) the Aggregate Type Name associated with this aggregate;
- iii) a boolean field, IsActive. If IsActive has the value "True", then the aggregate is known to be currently active (stored) in at least one system. If this value is "False", then the aggregate is not known to be stored in any system;
- iv) for each system S in which this aggregate is known to be stored, the following additional information is stored: the record IDs of the records in which this aggregate is stored in S, the name of the system S, and an optional "cookie", which contains system specific information related to this aggregate in system S. (As discussed below, this system information cookie may contain, for example, additional information for the system to use to facilitate mapping between the record ID and its internal storage). Note that this table allows one to associate with any single aggregate, for any system S, a set of Record IDs for this aggregate. This may be a single Record ID if S stores the data properties of the aggregate in a single record structure, or multiple record IDs if it stores these properties in multiple record structures. Note also that it is possible for a single Record ID for a system S to be associated with multiple aggregates. This would be the case, for instance, if a single record in system S stored properties associated with multiple aggregates of the data model. However, we place the following restriction: for any system S, there exists only one aggregate in which S associates the particular set of system ID(s) Id-1, Id-2, ..., Id-n. That is, system S may associate an particular ID with multiple aggregates as part of "key" for the aggregate. But each set of system IDs for any given aggregate must be unique in order to enable the hub to uniquely identify the aggregate given a particular set of system specific IDs. However, see below for cases where it is possible to relax this rule.

We assume that a particular entry in xRef table can be found in one of two ways: First, given a GUID, the unique aggregate associated with this GUID can be retrieved from the table. Second, given a system name and a set of system specific IDs associated with a particular aggregate in this system, the unique aggregate associated with these IDs can be retrieved from the table.

There exists one logical xRef table in the integration hub, although it may be partitioned into separate physical implementations for various reasons such as performance. A graphical view of the xRef table is given in figure 1:

GUID	Aggregate Type Name	IsActive
Record IDs for System1	System1 Name	SystemInfoCookie1

Record IDs for System2	System2 Name	CookieSystemInfo2
...

3. Aggregate Keys

A key data structure associated with an aggregate A is represented by four XML elements, a GUID element, an AGGREGATE TYPE NAME element, a SYSTEM DATA element, and an ACTION CODE element. The SYSTEM DATA element contains a System Name element, a list of Record ID elements, and optionally a system specific SystemInfo element *Cookie*. The XML structure of an Aggregate Key is given in figure 2:

```

<KEY>
  <GUID>
  <\GUID>
  <AGGREGATE TYPE NAME>
  <\AGGREGATE TYPE NAME>
  <SYSTEM DATA>
    <SYSTEM NAME>
    <\SYSTEM NAME>
    <IDENTIFIERS>
      <ID>
      <\ID>
      <ID>
      <\ID>
      ...
    <\IDENTIFIERS>
    <COOKIESYSTEMINFO>
    <\COOKIESYSTEMINFO>
  <\SYSTEM DATA>
  <ACTION CODE>
  <\ACTION CODE>
<\KEY>

```

We assume that everytime an aggregate is sent in a message, this key structure is sent with it, although some elements of the structure may or may not be present in any particular message, as described below. The <GUID> is a globally unique ID created by the hub to uniquely identify this aggregate. The <Aggregate Type Name> is the type name of this aggregate, as specified by the data model. The <SYSTEM NAME> element and the <IDENTIFIERS> element contain the internal system IDs associated with this aggregate in the particular system known by the given system name. The <COOKIE SYSTEMINFO> element is an uninterpreted string of information that a particular system wants to store with this <GUID>. The <ACTION CODE> can have the following values: "Created", "Exists", "Deleted", "AttachID", or "ModifyID". The use of these ACTION CODEs in a message are described below.

III. Mapping system specific IDs to aggregate GUIDs

Describe here the various cases of one-to-one, one-to-many, many-to-one, and many-to-many mappings between system records and aggregates.

IV. Algorithms for maintaining the xRef table and for substituting the appropriate IDs as messages are sent to and from the integration hub

Every message sent through the hub operates on aggregates. The semantics of each message determines whether any particular aggregate in the message is a newly created aggregate (has never been seen before by the hub, and therefore did not have a GUID associated with it prior to this message invocation), is an existing aggregate (which the hub has already created a GUID for) in which the

message is simply "operating on" (e.g., may be changing semantic content of the data properties, or may be asking to read the data associated with this aggregate in another system, or may be required for an arbitrary business action as determined by the semantics of the message), or is an existing aggregate (for which the hub has already created a GUID) being deleted from a particular system. Finally, a message may be sent to the hub by a system asking it to associate its system specific ID(s) with an existing aggregate or to update the system specific ID(s) it has already associated with a specific aggregate. Hence we describe the actions performed by the sending system, by the hub, and by the receiving system, for each of these cases. Since any message may have multiple aggregates, this describes the actions taken by the systems and hub for every aggregate in a message.

For any message the hub receives, the semantics of the message will usually dictate that the message is intended for one or multiple other systems. The intended recipient(s) of the message may be explicitly stated in the header of the message, or it may be inferred by the hub. For instance, the hub may utilize a publish and subscribe algorithm, in which the sender sends the message to the hub, and the hub forwards the message to all systems that have previously indicated interested in receiving messages of this nature. We assume that the hub utilizes some mechanism for determining which systems it is to send (forward) messages it receives, but the exact mechanism used is independent of mechanism described in this patent.

Case 1: The sending system is creating a new aggregate.

In this case, the sending system would leave the GUID element empty, but include the AGGREGATE TYPE NAME and any system IDENTIFIERS it wants to associate with this aggregate. It must associate at least one IDENTIFIER with this aggregate. It would include its SYSTEM NAME, and optionally add any private information it wants to associate with this aggregate in the SYSTEMINFOCOOKIE element. The ACTION CODE for this aggregate would be set to "Created".

The hub, upon receiving this message, would create a new GUID for this aggregate, and create a new entry in xRef table. This entry would have this GUID in its GUID field, and its IsActive field would be set to TRUE. It would take the SYSTEM NAME, IDENTIFIERS, and ~~COOKIE~~SYSTEMINFO contained in the KEY structure and enter these into this xRef table entry. Finally, the hub would forward this message to the intended recipient(s) of the message. When it forwards the message, it keeps the KEY structure the same as how it was received, with the following exceptions: it deletes the SYSTEM IDENTIFIERS (so that the IDENTIFIERS element becomes EMPTY), sets the SYSTEMINFO~~COOKIE~~ and SYSTEM NAME elements to be empty, and sets the GUID to be the value of the newly created GUID.

Each receiving system decides whether or not it wants to locally store a copy of this aggregate. If so, it will create system record(s) containing the data properties associated with this aggregate. It can either internally associate the GUID with these system record(s), or it can use its own system specific ID(s) for this record. In the latter case, it needs to inform the hub of the value of these ID(s) to be used in future messages. It does so by sending a special AttachID message to the hub. This is described in case 4.

Case 2: The sending system is operating on an existing aggregate

In this case, the sending system either knows the GUID associated with the aggregate it is operating on, or has system specific ID(s) associated with this aggregate. In the former case, it would set the <GUID> element to the GUID for this aggregate and leave the <IDENTIFIERS> element empty. In the latter case, it would leave the GUID element empty but send its system specific ID(s) in the <IDENTIFIERS> element. Since the hub already knows the type of this aggregate, the sending system may optionally leave the <AGGREGATE TYPE NAME> element empty. The <SYSTEMINFO~~COOKIE~~> element should be left empty, and the <ACTION CODE> must be set to "Exists".

The hub, upon receiving this message, would determine which systems to forward this message to. Since this aggregate already exists, there is an entry for this aggregate in the xRef table. If the sending system sent the GUID for this aggregate, the hub can retrieve the xRef entry E for this aggregate by using this unique GUID. Otherwise, the sending system sent system specific IDs for this aggregate. As

dictated above, this means that the these IDs uniquely identify the aggregate, and the hub can use these IDs to retrieve the xRef entry E for the aggregate associated with these IDs.

For each system S to which the hub needs to forward this message, S may or may not already have an entry associated with this aggregate in the xRef entry E. First consider those systems S which already have an entry associated with E. In this case, the hub looks up the entry for this system associated with E, retrieving the Record ID(s) and ~~Cookie~~system information (SYSTEMINFO, if present) for this system. The hub forwards this message to S, sending the KEY structure with the GUID and AGGREGATE TYPE NAME set according to the entry in the xRef table. The SYSTEM NAME, IDENTIFIERS, and SYSTEMINFO~~COOKIE~~ would be set according to the information given in entry E. The ACTION CODE would be set to "Exists". When S receives this message, because the action code is "Exists" and because its system specific ID(s) are included in the KEY structure, it can find those system record(s) associated with this aggregate, and operate on the record(s) as dictated by the semantics of the message. If needed, it may use the ~~COOKIE~~SYSTEMINFO to facilitate its ability to locate the records associated with this aggregate.

Second consider those systems S which the hub needs to forward this message to but do not have an entry associated with E. In this case, the hub forwards the message to S, sending the KEY structure with the GUID and AGGREGATE TYPE NAME set according to the entry in the xRef table. The SYSTEM NAME, is set to the system being sent to, but the IDENTIFIERS and SYSTEMINFO~~COOKIE~~ elements are set to be empty. The ACTION CODE would be set to "Exists". When S receives this message, because the action code is "Exists" and because no system specific ID(s) are included in the KEY structure, it knows that the aggregate being operated upon has been created previously in other systems but it has no associated SYSTEM IDs associated with this aggregate. The system may be able to use the GUID and AGGREGATE TYPE NAME to identify an existing system record(s) corresponding to this aggregate, may want to create new system specific record(s) for this aggregate, or may take some other action. What this specific system does upon receiving this message is system specific and is not dictated by the mechanism of this patent. However, if the system decides after receiving this message to associate system specific records with this aggregate, and to use system specific ID(s) for these records, it needs to inform the hub of the value of the system specific record ID(s) to be used in future messages regarding this aggregate. It does so by sending a special AttachID message to the hub. This is described in case 4.

Case 3: The sending system is deleting an existing aggregate from its internal system
In this case, the sending system either knows the GUID associated with the aggregate it is operating on, or has system specific ID(s) associated with this aggregate. In the former case, it would set the <GUID> element to the GUID for this aggregate and leave the <IDENTIFIERS> element empty. In the latter case, it would leave the GUID element empty but send its system specific ID(s) in the <IDENTIFIERS> element. Since the hub already knows the type of this aggregate, the sending system may optionally leave the <AGGREGATE TYPE NAME> element empty. The <SYSTEMINFO~~COOKIE~~> element should be left empty, and the <ACTION CODE> must be set to "Deleted".

The hub, upon receiving this message, would determine which systems to forward this message to. Since this aggregate already exists, there is an entry for this aggregate in the xRef table. If the sending system sent the GUID for this aggregate, the hub can retrieve the xRef entry E for this aggregate by using this unique GUID. Otherwise, the sending system sent system specific IDs for this aggregate. As dictated above, this means that the these IDs uniquely identify the aggregate, and the hub can use these IDs to retrieve the xRef entry E for the aggregate associated with these IDs. Since the sending system S has deleted this aggregate from its system, the hub can remove from E the entry associated with S (if any). If after removing this entry, there is no longer any system records associated with E, then the hub sets the IsActive field to "FALSE". Once an entry in the xRef table has its IsActive field set to false, it is subject to removal. Whether or not it gets removed from the table, and when this removal occurs, depends upon table management and archival processes.

The hub would then forward the message to each intended recipient system S, setting the KEY structure in the same manner as done above in case 2. The only exception is that the ACTION CODE element would be set to "Deleted".

Case 4: A system wants to associate its internal IDs with an existing aggregate, ~~or to update its internal IDs already associated with an aggregate~~

A system, usually -- but not exclusively -- at the point in time of learning of the creation of a new aggregate with a given GUID, can create a replicated version of the aggregate and tell the hub to associate its system record ID(s) with this aggregate. It may do so in a message responding to the message in which it was informed about this new aggregate, or may do so in a special message intended only for the hub. In either case, it would create a KEY structure as follows: it sets the GUID element to the GUID of the aggregate, and may optionally set the AGGREGATE TYPE NAME element. It sets the SYSTEM NAME element to its own system name, sets the IDENTIFIER element to contain the ID(s) of its records corresponding to this aggregate, and optionally sets the ~~COOKIE~~ SYSTEMINFO element to contain any private information it wants associated with this aggregate. It sets the ACTION CODE to "AttachID". Upon receiving this KEY structure, the hub inserts a new row into the xRef table entry associated with this aggregate (which it identifies by the GUID element in the KEY structure) by putting the SYSTEM NAME, IDENTIFIERS, and SYSTEMINFO~~COOKIE~~ contained in the KEY structure into the appropriate fields of this new row.

(Move the following paragraph to Case 5 below)

Once a system has requested that the hub associate system specific ID(s) with a particular aggregate, it may later wish to change the ID(s) associated with that aggregate. This may be do, for instance, to reorganization of the system specific data within the host system. In this case the system would send a message to the hub containing a modified KEY structure. This KEY structure would be identical to the one described in section II above, with the following exceptions: instead of an IDENTIFIERS element, it would contain an <OLD IDENTIFIERS> element and a <NEW IDENTIFIERS> element. Each of these elements would themselves contains a repeating <ID> element (i.e., one or more ID(s)). The sending system would optionally leave the GUID and AGGREGATE TYPE NAME elements empty, would set its name to the SYSTEM NAME element, set the <OLD IDENTIFIERS> element to contain the ID(s) it previously associated with this aggregate, and set the <NEW IDENTIFIERS> element to contain the new ID(s) it wants associated with this aggregate. It would optionally set the ~~COOKIE~~ SYSTEMINFO element to contain any private information it wants to associate with this aggregate. The ACTION CODE for this aggregate would be set to "~~Attach~~ModifyID". Upon receiving this KEY structure, the hub first retrieves the xRef entry E associated with this aggregate, which it would find using either the GUID, if supplied, or the system ID(s) contained in the <OLD SYSTEM IDENTIFIERS> element. It would then replace the row in E allocated to this system with the a new row, by putting the SYSTEM NAME, NEW IDENTIFIERS, and SYSTEMINFO~~COOKIE~~ contained in the KEY structure into the appropriate fields of this new row.

Case 5: A system wants to update its internal IDs already associated with a particular aggregate

V. Enhancements for supporting creation & deletion requests

The algorithms, as well as the data structures, described above require that systems indicate to the hub when an aggregate is created or deleted from a particular system. In reality, sometimes a system may want to request the creation of a new aggregate, but the aggregate will not get created until another different system confirms the request with a positive response. For instance, one system (say, a call center application) may want to create a new customer record, but the "system of truth" for customer information may have to be consulted before such a record is actually created (for instance, it may be that this customer already exists in a different record unknown to the call center application).

There is nothing in the algorithms and data structures that prevents this scenario from an application

perspective. Any message, including a "CreateCustomerRecordRequest" message can be exchanged between systems. However, from the hub's perspective, this scenario cannot use the algorithm given above in IV, case 1), since that would cause the hub to create a new data structures (e.g., xRef table entry, etc.) for the aggregate when the initiating system sends the creation request to the destination system, even though the destination system may respond and tell the initiating system not to create the new aggregate. A similar situation applies to deletion request.

To handle these scenarios, we introduce a new action code "Passthrough". When the system requesting that a new aggregate be created, it formats the key structure in a fashion identical to that of the sending system in IV, case 1, except that it sets the ACTION CODE to "Passthrough" instead of "Created". The hub, when receiving this message, defers creating any xRef structure, but just passes the message, including the key structure created by the initiating system, to the destination system. The destination system will respond to this message by either accepting or rejecting the creation of the aggregate. In the latter case, the hub just returns the message to the initiating system. In the former case (when the destination system accepts the creation of the aggregate), the destination system attaches the initial key request onto the response, with the following modifications: it changes the ACTION CODE from "Passthrough" to "Created". Assuming the destination system also wants to associate its own system identifiers to this newly created aggregate, it would also create another SYSTEM DATA element in the key, containing its own SYSTEM NAME, SYSTEM IDENTIFIER(s), and optionally SYSTEM INFO. (Hence the key structure returned to the hub would contain two SYSTEM DATA elements, one with information about the sending system and one with information about the destination system).

Upon receiving this message, the hub then creates a new GUID and xRef entry for this new aggregate, similar to steps described in IV, case 1. However, this time it would create two system entries for this aggregate, one for the initiating system and one for the destination system. It would then inform both the initiating and destination systems about the GUID associated with this aggregate.

A potential optimization on this algorithm would have the hub optimistically create a GUID and xRef entry for this aggregate before forwarding the original message to the destination system, and then undo these actions if the destination system rejects the creation of the aggregate. This optimization would probably be worthwhile if in the vast majority of the cases the destination system accepts the creation request.

Deletion requests can be handled in a similar fashion.

(Form Revised 12/17/97)